Framework for adaptive fluid-structure interaction with industrial applications

Johan Jansson

KTH Royal Institute of Technology,
Department of High Performance Computing
and Visualization (HPCViz),
CSC KTH 100 44 Stockholm, Sweden
and
Basque Center for Applied Mathematics (BCAM),
Alameda de Mazarredo 14,
48009 Bilbao, Bizkaia Basque-Country, Spain
E-mail: jjan@kth.se

Niyazi Cem Degirmenci* and Johan Hoffman

KTH Royal Institute of Technology,
Department of High Performance Computing
and Visualization (HPCViz),
CSC KTH 100 44 Stockholm, Sweden
E-mail: ncde@kth.se
E-mail: jhoffman@kth.se
*Corresponding author

Abstract: We present developments in the Unicorn-HPC framework for unified continuum mechanics, enabling adaptive finite element computation of fluid-structure interaction, and an overview of the larger FEniCS-HPC framework for automated solution of partial diffential equations of which Unicorn-HPC is a part. We formulate the basic model and finite element discretisation method and adaptive algorithms. We test the framework on a 2D model problem consisting of a flexible beam in channel flow, and to illustrate the capabilities of the computational framework, we show two application examples from industry and medicine. We simulate a flexible mixer plate in turbulent flow in an exhaust system where the target output is aeroacoustic quantities. The second example is a self-oscillating vocal fold configuration, where the ultimate goal is to predict how the voice is affected by physiological changes from aerodynamics. Here we give the displacement signal of a point on the folds.

Keywords: high performance computing; fluid structure interaction; adaptive mesh refinement.

Reference to this paper should be made as follows: Jansson, J., Degirmenci, N.C. and Hoffman, J. (2013) 'Framework for adaptive fluid-structure interaction with industrial applications', *Int. J. Materials Engineering Innovation*, Vol. 4, No. 2, pp.166–186.

Biographical notes: Johan Jansson is leading the research line Computational Technology at the Basque Center for Applied Mathematics which has the aim of developing automated and adaptive methods and the open source FEniCS-HPC software framework (fenicsproject.org) for solving partial differential equations based on FEM with good scaling on massively parallel computers. A particular focus is on turbulent flow and fluid-structure interaction, and he also develops applications in aerodynamics and biomechanics. He received his PhD from Chalmers University of Technology in Sweden and during his post-doc at KTH Royal Institute of Technology was part of starting the Computational Technology Laboratory. He now has research positions at KTH and BCAM.

Niyazi Cem Degirmenci is a PhD student at Royal Institute of Technology, at the Department of High Performance Computing and Visualization. His research interests include numerical methods for fluid structure interaction, modelling and simulation of human vocal folds, development of the software package Unicorn, a unified continuum mechanics solver and mesh adaptation algorithms.

Johan Hoffman is a Professor in Numerical Analysis, Deputy Head of the Department of High Performance Computing and Visualization, and the Principal Investigator of the Computational Technology Laboratory. His main research interests are computational mathematical modelling with differential equations and high performance adaptive finite element methods, turbulent flow and fluid-structure interaction, with applications in aerodynamics, aeroacoustics, biomedicine and geophysics. He is one of the founders of the FEniCS open source software project, with in particular the DOLFIN and Unicorn simulation tools. He was selected for a research fellowship of the Swedish Research Council in 2010. In 2008 he was awarded a European Research Council starting grant, and an individual grant for the advancement of research leaders (Framtidens Forskningsledare) by the Swedish Foundation for Strategic Research. He is the recipient of a Leslie Fox Prize in numerical analysis in 2005, and a Bill Morton Prize on computational fluid dynamics in 2001.

This paper is a revised and expanded version of a paper entitled ‘An adaptive finite element method for fluid structure interaction simulation of vocal folds’ presented at Young Investigators Conference – YIC2012, Aveiro, Portugal, April 2012.

1 Introduction

This paper gives an overview of a framework for fluid-structure interaction (FSI) simulation with the possibility of adaptive mesh refinement (AMR) based on solving a dual problem giving sensitivity information in an error estimate. The framework is based on incompressible unified continuum (UC) modelling where we define balance laws for mass and momentum for the continuum and keep a stress σ and phase variable θ as data for defining properties of the continuum, such as constitutive laws and material parameters. The model is discretised with a stabilised finite element method (FEM)

giving a monolithic method for the whole FSI system. This allows the formulation of a residual and derivation of an error estimate for the whole fluid-structure system using standard techniques.

We ultimately target industrial and medical applications and the ability to predict mean quantities in turbulent flow is a key goal. In this paper we demonstrate:

1 an application in exhaust systems where we model a flexible mixer plate in channel flow

2 an application in voice generation where we model self-oscillating vocal folds given a static lung pressure

3 a 2D model problem with a flexible beam in channel flow where we apply the adaptive methodology to the full FSI problem.

1.1 UC model and AMR

As computational methods and hardware become more powerful it is now possible to model and simulate phenomena of increasing complexity. Traditionally simplifications have been made when modelling mechanical systems and only the fluid or solid part separately has been considered in models, or simple linear models have been used. Recently simulation frameworks for full non-linear FSI models have been developed and used to attack various problems of industrial and medical relevance (Bazilevs et al., 2006; Tezduyar et al., 2006).

In a FEM framework the problem is described by a mesh. AMR is the process of refining only the cells of the mesh (triangles/tetrahedra) necessary to satisfy a tolerance on the error quantity (Eriksson et al., 1995; Becker and Rannacher, 2001; Giles and Süli, 2002; Babuška, 1986). AMR is based on an a posteriori error estimate where the error quantity of the solution to the partial differential equation (PDE) is estimated in terms of the residual of the equation (which is computable). The error estimate can then be broken down as a sum over all the cells in the mesh, and the cells with the largest contribution to the error can be identified and refined.

To be able to control the discretisation error in a simulation, it is necessary to

1 have knowledge of the sensitivity of the error quantity with regard to the mesh size

2 be able to refine the cells which contribute to the sensitivity.

AMR allows us to control the discretisation error *efficiently* in the sense that we only need to refine the cells which contribute the most, and leave the rest of the cells untouched. Non-AMR, uniform mesh refinement, is prohibitively expensive in 3D, and for many problems AMR is the only way to make the problems tractable at all.

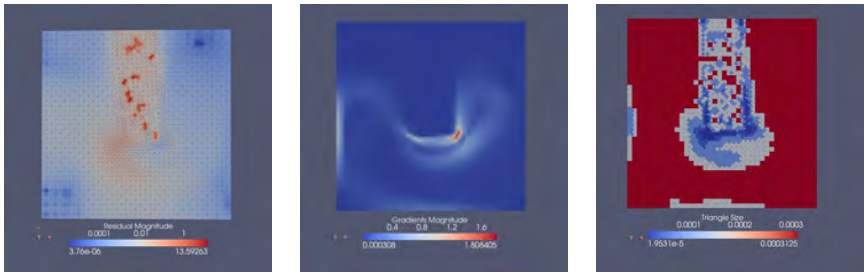
In this paper we describe the Unicorn framework for modelling and simulating FSI problems of industrial and medical relevance with AMR. The framework consists of a UC model describing the conservation equations for the whole system, a unified Cauchy stress σ for all phases and a phase function θ. The FSI problem is thus treated as a multiphase flow problem, where a phase function θ identifies the solid and fluid, respectively. Typically we let the finite element mesh track the solid deformation and thus a piecewise constant phase function will not cut any elements in the mesh, but

will stay with the solid throughout the computation, and thus no equation for the phase variable needs to be solved.

The UC model allows us to derive a residual and error estimate for the whole FSI system, and thus an AMR algorithm, which is described in detail further on in the paper.

The a posteriori error estimate is based on introducing a linearised dual problem, and expressing the error in a given output as a combination of a *residual* of the computed solution and derivatives of the *dual solution*, acting as weights, giving information about the effect of the residual on the output error. See Figure 1 for a basic illustration of the error estimation, the details are presented in later sections.

Figure 1 Basic illustration of the error estimate consisting of the product of the *residual* and gradient of the *dual solution* (a) residual magnitude (b) dual velocity gradient magnitude (c) adaptively refined mesh (see online version for colours)



(a) (b) (c)

Notes: The product is computed for each cell and gives an estimate of the error contribution for that cell. Where it is high, the mesh is refined, giving an adaptively refined mesh right (starting from a regular mesh).

1.2 Automated high performance computation in the FEniCS-HPC framework

FEniCS-HPC is an open source framework for automated solution of PDE on massively parallel architectures, providing automated evaluation of variational forms given a high-level description in mathematical notation, duality-based adaptive error control, implicit turbulence modelling by use of stabilised FEM and strong linear scaling up to thousands of cores (Hoffman et al., 2011b, 2012a; Jansson et al., 2012; Kirby, 2012; Logg et al., 2012b; Hoffman et al., 2012b, 2012c). FEniCS-HPC is a branch of the FEniCS (2003) framework focusing on high performance on massively parallel architectures.

The framework is based on components with clearly defined responsibilities. The main components are the following, with their dependencies shown in the dependency diagram in Figure 2:

- automated generation of finite elements/basis functions (FIAT)

  $$e = (K, V, \mathcal{L})$$

- automated evaluation of variational forms on one cell based on code generation (FFC)

$$A^K = a_K(v, U) = \int_K \nabla v \cdot \nabla U dx = \int_K R(v, U) dx$$

- automated high performance assembly of discrete systems (DOLFIN-HPC)

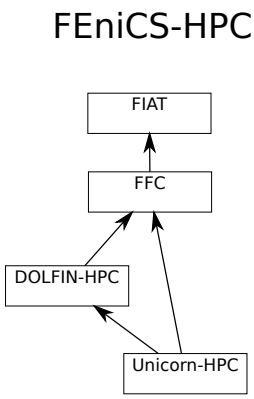
$$\begin{aligned} &A = 0 \\ &\texttt{for all cells } K \in \mathcal{T}_\Omega \\ &\quad A \texttt{ += } A^K \end{aligned}$$

- automated UC modelling (Unicorn-HPC)

$$R_{UC}(v, W) = (v, \rho(\partial_t u + (u \cdot \nabla)u) + \nabla \cdot \sigma - g) + SD(v, W).$$

where K is a cell in a mesh $\mathcal{T}$, V is a finite-dimensional function space, $\mathcal{L}$ is a set of degrees of freedom and A is a matrix.

Figure 2 FEniCS-HPC component dependency diagram



Unicorn-HPC is solver technology (models, methods, algorithms and software) with the goal of automated high performance simulation of realistic continuum mechanics applications, such as drag or lift computation for fixed or flexible objects (FSI) in turbulent incompressible or compressible flow. The basis for Unicorn is Unified Continuum (UC) modelling (Hoffman et al., 2011c) formulated in Euler (laboratory) coordinates, together with a General Galerkin (G2) adaptive stabilised finite element discretisation with a moving mesh for tracking the phase interfaces. Unicorn formulates and implements the adaptive G2 method applied to the UC model, and interfaces to other components in the FEniCS-HPC chain (FIAT, FFC, DOLFIN-HPC) described above.

2 UC FSI

The incompressible UC model takes the form:

$$\begin{array}{ll} \rho(D_t u_i + \theta u_j D_{x_j} u_i) = D_{x_j}\sigma_{ij} + f_i \\ D_{x_j} u_j & = 0 \\ D_t\theta + D_{x_j}(u_j\theta) & = 0 \end{array} \tag{1}$$

where u is velocity with components u_i, σ is the stress tensor, f is the external force and θ is the phase function that marks the continuum as either fluid or solid ($\theta := 1$ for fluid and $\theta := 0$ for solid). The three equations describe conservation of momentum, conservation of mass and phase convection respectively.

An example of fluid-structure constitutive laws can take the form:

$$\begin{array}{ll} \sigma & = \sigma_D - pI \\ \sigma_D & = \theta\sigma_f + (1-\theta)\sigma_s \\ \sigma_f & = 2\mu_f\epsilon(u) \\ D_t\sigma_s & = 2\mu_s\epsilon(u) + \nabla u\,\sigma_s + \sigma_s\nabla u^\top \end{array} \tag{2}$$

where σ_D stands for deviatoric stress, p for pressure, σ_f and σ_s for fluid and solid stresses, μ_f for viscosity, μ_s for the shear modulus of the structure, which is the ratio of the shear stress to shear strain. The strain rate tensor ϵ is defined as $\epsilon(u) = \frac{1}{2}(\nabla u + \nabla u^\top)$. In 2D, the result of applying the operator ∇ on $u = (u_0, u_1)$ is the tensor which is equal to the dyadic product of (D_{x_0}, D_{x_1}) and (u_0, u_1).

The reader is referred to Hoffman et al. (2011c) for the complete derivation of fluid-structure constitutive laws for Neo-Hookean solid, which fundamentally results from considering the time derivative of the displacement in elasticity equations.

2.1 G2 finite element discretisation

Our computational approach is based on stabilised FEMs, together with adjoint-based adaptive algorithms, and residual-based implicit turbulence modelling and shock capturing, for related work see, e.g., Bazilevs et al. (2007), Guasch and Codina (2007) and Guermond et al. (2011).

The G2 method for high Reynolds number flow, including turbulent flow and shocks, takes the form of a standard Galerkin finite element discretisation together with

1 least squares stabilisation of the residual

2 residual-based shock capturing.

With a G2 method, we define turbulent flow as the non-smooth parts of the flow where the residual measured in L_2-norm increases as the mesh is refined, whereas in a negative H^{-1}-norm the residual decreases with mesh refinement (Hoffman and Johnson, 2008). That is, we characterise turbulence by a pointwise large residual which is small in average, corresponding to the equations being satisfied only in a mean value sense, which is sufficient to approximate mean value quantities of a turbulent flow field using G2.

We split the time interval I into subintervals $I_n = (t_{n-1}, t_n)$, with associated space-time slabs $S_n = \Omega \times I_n$, over which we define space-time finite element spaces, based on a spatial finite element space W^n over a spatial mesh $\mathcal{T}_n$ (Hoffman and Johnson, 2007).

In a cG(1)cG(1) method (Hansbo, 2000; Hoffman et al., 2011a) we seek an approximate solution $\hat{U} = (U, P)$ which is continuous piecewise linear in space and time. With W^n a standard finite element space of continuous piecewise linear functions, and W_0^n the functions in W^n which are zero on the boundary Γ, the cG(1)cG(1) method for constant density incompressible flow with homogeneous Dirichlet boundary conditions for the velocity takes the form: for $n = 1, ..., N$, find $(U^n, P^n) \equiv (U(t_n), P(t_n))$ with $U^n \in V_0^n \equiv [W_0^n]^3$ and $P^n \in W^n$, such that

$$\begin{aligned} &((U^n - U^{n-1})k_n^{-1} + (\bar{U}^n \cdot \nabla)\bar{U}^n, v) + (2\nu\epsilon(\bar{U}^n), \epsilon(v)) \\ &- (P^n, \nabla \cdot v) + (\nabla \cdot \bar{U}^n, q) + SD_\delta^n(\bar{U}^n, P^n; v, q) = (f, v) \\ &\forall \hat{v} = (v, q) \in V_0^n \times W^n \end{aligned} \tag{3}$$

where $\bar{U}^n = 1/2(U^n + U^{n-1})$ is piecewise constant in time over I_n, with the stabilising term

$$\begin{aligned} SD_\delta^n(\bar{U}^n, P^n; v, q) \equiv (\delta_1(\bar{U}^n \cdot \nabla \bar{U}^n + \nabla P^n - f), \bar{U}^n \cdot \nabla v + \nabla q) \\ + (\delta_2 \nabla \cdot \bar{U}^n, \nabla \cdot v), \end{aligned}$$

where we have dropped the shock capturing term, and where

$$\begin{aligned} (v, w) &= \textstyle\sum_{K \in \mathcal{T}_n} \int_K v \cdot w \, dx, \\ (\epsilon(v), \epsilon(w)) &= \textstyle\sum_{i,j=1}^3 (\epsilon_{ij}(v), \epsilon_{ij}(w)), \end{aligned}$$

with the stabilisation parameters

$$\begin{aligned} \delta_1 &= \kappa_1 (k_n^{-2} + |U^{n-1}|^2 h_n^{-2})^{-1/2} \\ \delta_2 &= \kappa_2 h_n \end{aligned}$$

where κ_1 and κ_2 are positive constants of unit size. For turbulent flow we choose a time step size

$$k_n \sim \min_{x \in \Omega} (h_n / |U^{n-1}|).$$

We note that the least squares stabilisation omits the time derivative in the residual, which is a consequence of the test functions being piecewise constant in time for a cG(1) discretisation of time (Hoffman et al., 2011a).

For variable density incompressible flow and FSI the method takes a similar form, although in the first case we include a cG(1)cG(1) discretisation of the equation for conservation of mass, and in the second case we add an equation for the structure stress.

For the UC FSI model, we introduce a piecewise constant solid stress term S_s, and the mesh motion adds an ALE mesh velocity β_h to the convective velocity:

$$\begin{aligned} &(\rho((U^n - U^{n-1})k_n^{-1} + ((\bar{U} - \beta_h)^n \cdot \nabla)\bar{U}^n), v) \\ &+ (1 - \theta)(S_s, \nabla v) + \theta(2\mu_f \epsilon(\bar{U}^n), \epsilon(v)) - (P^n, \nabla \cdot v) + (\nabla \cdot \bar{U}^n, q) \\ &+ SD_\delta^n(\bar{U}^n, P^n; v, q) = (f, v) \, , \, \forall \hat{v} = (v, q) \in V_0^n \times W^n \end{aligned}$$

3 State of the art

A posteriori error estimation and AMR in the context of fluid structure interaction problems have been studied previously in Dunne and Rannacher (to appear) and van der Zee et al. (2010). In Dunne and Rannacher (to appear), a Eulerian framework simplifies error estimation, but the use of level set type methods causes difficulties with an interface which is crossing cells. For FSI problems where thin elastic bar is immersed in incompressible fluid with inflow, no duality-based refinement convergence results are presented. For these problems, heuristic refinement approach concerning the distance to solid is applied and convergence results are presented accordingly. The paper does show convergence results for duality-based refinement for stationary elastic flow cavity problem and the CSM test where elastic bar is immersed in incompressible fluid with no inflow but subject to gravity.

In van der Zee et al. (2010) two approaches for linearisation of primal problem to construct the dual problem are presented: In the domain-map linearisation approach, the fluid subproblem is first transformed to a fixed reference domain and linearisation is done with respect to the domain transformation map. In the shape-linearisation approach, fluid unknowns have a fixed configuration, and shape-derivative techniques are used for linearisation of the weak form of the fluid subproblem. The dual problems correspond to coupled fluid-structure problems that are constructed from adjoints of the linearised problems with nonstandard coupling conditions. In the paper, three dimensional problems or problems with deformable solids are not considered. In such extensions, the main challenge becomes the derivation of the coupling conditions for the dual problem, as stated by the author.

4 A posteriori error estimation

In most applications, controlling the error in some quantity of output is of major concern. Examples for such quantities of interest can be drag, lift, pressure differences or displacements. Based on the a posteriori error estimates we construct an algorithm for AMR with respect to the error in the chosen output. Babuška (1986) and Babuška and Miller (1984) were the pioneers to use duality arguments in the context of postprocessing 'quantities of physical interest' for elliptic problems. Our work is based on the general framework developed by Eriksson and Johnson (1988) and Eriksson et al. (1995, 1996) and Becker and Rannacher (2001, 1996) with co-workers.

The discretisation of the primal and dual problems are done by a Galerkin FEM, based on variational forms of (1) and (2).

We here recall the basic steps of a posteriori error estimation based on this framework. The Galerkin method for an abstract problem $A(u) = f$ on Ω where A is a linear operator is:

find $u \in V^h$, such that

$$(A(u), v) = (f, v) \qquad \forall v \in V_0^h \tag{1}$$

where $(.,.)$ is an inner product, and $\Omega \subset \mathbb{R}^d$ is the spatial domain, V is a Hilbert space, U is the calculated solution in $V^h \subset V$ the discrete finite element subspace of functions constructed on a space-time mesh with size h. $V_0^h \subset V^h$ consists of functions that vanish where Dirichlet boundary conditions are specified.

The basic idea for bounding the error in a functional such as drag starts with making use of Riesz' representation theorem for bounded functionals

$$|M(u-U)| = |(u-U,\zeta)|$$

M is the functional we are interested in, such as a pressure difference in different areas of domain, and u is the exact solution.

We construct a dual problem:

$$A^*(\phi) = \zeta \tag{2}$$

with boundary conditions that allows for any $v, w \in V$

$$(A(v), w) = (v, A^*(w)) \tag{3}$$

It is then possible to bound the error in a functional for computations on a space-time mesh with size h as

$$\begin{aligned}
|M(u-U)| &= |(u-U,\zeta)| \\
&= |(u-U, A^*(\phi))| && \text{using (2)} \\
&= |(A(u-U),\phi)| && \text{using (3)} \\
&= |(R(U),\phi)| \\
&= |(hR(U), h^{-1}(\phi - \pi_h\phi))| \\
&\leq |hR(U)||CD\phi|
\end{aligned}$$

where $R(U) = f - A(U)$ is the residual, π_h is an interpolation operator onto V_h, D is a space-time derivative and C is an interpolation error constant depending on the minimum angle of the mesh and order of basis functions (Eriksson et al., 2003). Note that we have used the Galerkin orthogonality from (1).

For non-linear problems, the usual procedure is using the linearised problem to obtain the dual problem.

Here it is necessary to mention that the initial condition for the dual problem is given at the final time T so that (3) holds. The residual can only be obtained after solving the primal problem (1) until time T, and to obtain ϕ the dual problem is solved backwards in time. The adaptive algorithm can be stated as follows:

1 solve the primal problem (1) until final time T_f

2 solve the dual problem which has initial conditions at T_f backwards until the initial time T_i

3 calculate the error indicators ε (details given in the Appendix) using the gradient of the solution of the dual problem and the residual of the primal problem

4 if the error bound is below the tolerance terminate

5 refine a specified percentage of the cells where the contribution to the error indicator is highest

6 return to step 1.

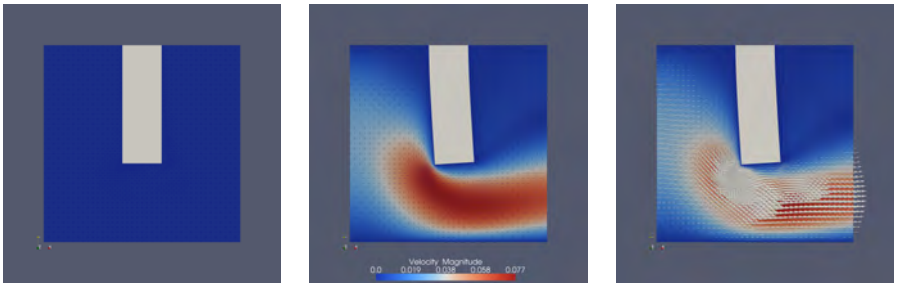
The construction of the dual problem such that (3) holds is tedious and error prone. Therefore we used a python tool which automates symbolic integration by parts operations to obtain the dual problem. In order to stabilise the dual problem, streamline diffusion type stabilisation terms (Hughes and Brooks, 1979, 1982; Johnson and Nävert, 1981) are added. Using the symbolic tool and interpreting its output is described in the Appendix. Necessary form files for implementation and the symbolic tool is available under http://www.csc.kth.se/ncde/IJMMME.tar

In our implementation we use mesh adaptation: local coarsening, refinement and swapping operations to maintain the mesh size/quality as described in Compère et al. (2010) instead of a method like global mesh smoothing. The primal and dual problems are solved on the same mesh, this means we need to store the primal mesh to be able to solve the dual problem. In order to avoid the large cost of saving meshes at each time step, the differences before and after a mesh adaptation process (Compère et al., 2010) are recorded and at each dual problem solving step the original mesh is recovered using this information. It is important to emphasise that using mesh adaptation enables us to save a small set of changes between two mesh configurations at consecutive timesteps, which would not be true with global mesh smoothing.

5 Benchmark for AMR

We have tested our implementation for the simple time dependent benchmark problem given in **?** where an elastic bar in a flow in 2D is simulated as in Figure 3.

Figure 3 Problem description for simple time dependent benchmark problem, (a) initial state (b) $t_{final} = 60$ velocity magnitude (c) $t_{final} = 60$ velocity vector (see online version for colours)

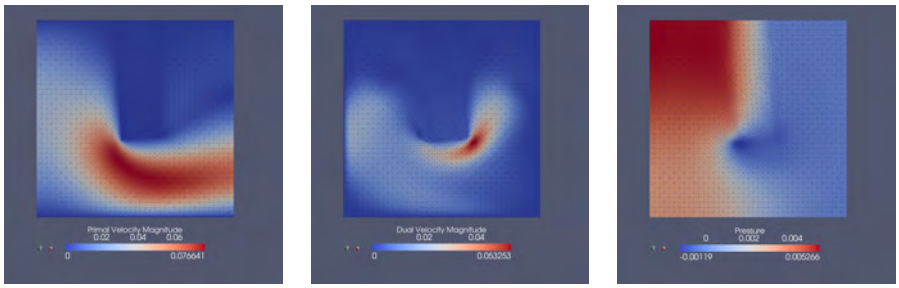


(a) (b) (c)

An inflow with parabolic velocity profile of $v_x = 0.03 * y * (1 - y), v_y = 0$ in a unit square domain with a fluid with viscosity $\nu_f = 0.001$ and an elastic bar with $\mu_s = 2$. The fluid and solid both have densities $\rho = 1$. The top and bottom boundaries have no-slip boundary condition while on the right boundary we have no pressure boundary condition. Primal and dual solutions (velocity, pressure and phase magnitude) for $t = 52$ for the finest adaptive mesh is given in Figures 4(a) to 4(f). The momentum residual at time $t = 52$ is shown at Figure 4(g). Figure 4(h) shows gradient of dual momentum residual which acts as weight for momentum residual to compute error contribution of the cells. The functional of interest is mean pressure difference in two coloured areas seen in Figure 4(d) during the whole simulation and between $t = 45$ and $t = 60$. The

relative error in this functional with respect to the finest mesh for different uniform and adaptive refinements with respect to number of vertices in mesh are given in Table 1. Fast convergence for the adaptive refinement results with respect to the finest uniform refinement can be observed immediately with much fewer number of elements. The set of Figure 5 show how the AMR been done starting from the initial mesh where we see the effect of jumps in the solid-fluid interface as well as the effect of dual weight in the cells near the corner where flow changes direction.

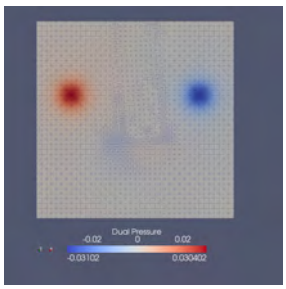
Figure 4 Numerical results for simple time dependent benchmark problem, (a) primal velocity magnitude for $t = 52$ (b) dual velocity magnitude for $t = 52$ (c) primal pressure for $t = 52$ (d) dual pressure for $t = 52$ (e) primal phase for $t = 52$ (f) dual phase for $t = 52$ (g) residual magnitude at $t = 52$ (h) dual velocity gradient magnitude for $t = 52$ (see online version for colours)



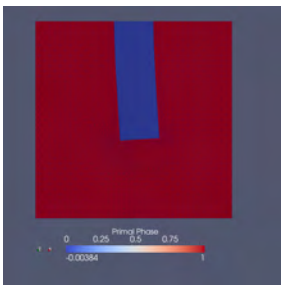
(a)

(b)

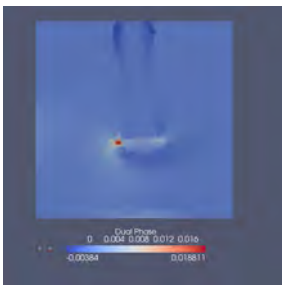
(c)



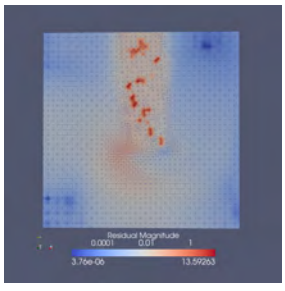
(d)



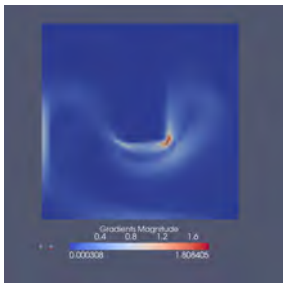
(e)



(f)



(g)

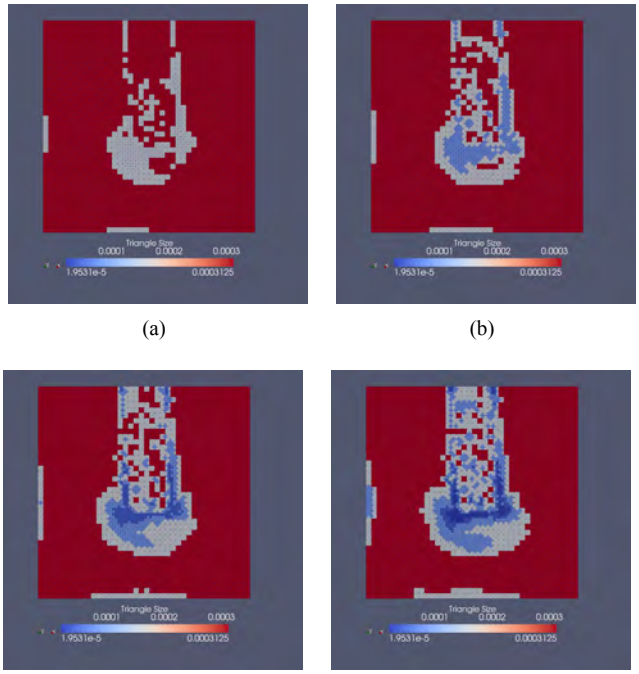


(h)

Table 1 Table of convergence wrt. four times uniformly refined mesh for functional between time $t = 45$ and $t = 60$

Refinement type	*Number of cells*	*Number of vertices*	*Relative error*
Initial mesh	3,200	1,681	0.05241
Uniform – level1	6,400	3,281	0.01252
Uniform – level2	12,800	6,561	0.00348
Uniform – level3	25,600	12,961	0.00317
Uniform – level4	51,200	25,921	(reference)
Adaptive – level1	3,588	1,875	0.01386
Adaptive – level2	4,152	2,158	0.00347
Adaptive – level3	4,760	2,462	0.00314

Figure 5 Initial meshes after different adaptive refinement steps, (a) first adaptive refinement (b) second adaptive refinement (c) third adaptive refinement (d) fourth adaptive refinement (see online version for colours)



(a) (b)

(c) (d)

6 Industrial and medical applications

Using the UC model we study applications with relevance in industry and medicine:

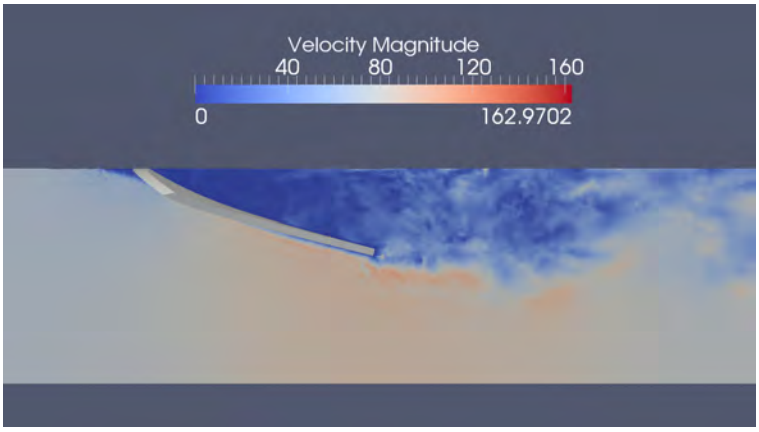
1 a *flexible mixer plate in an exhaust system* where the target is the aero-acoustical properties of the system

2 *self-oscillating vocal folds* driven by the lung pressure where the target is prediction of voice properties based on geometrical and mechanical changes in the tissue and a better general understanding of the mechanics.

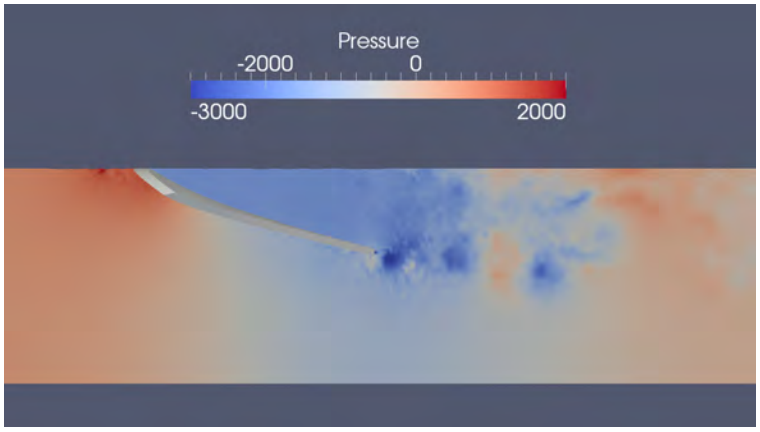
6.1 Flexible mixer plate in an exhaust system

An experimental configuration approximating an exhaust system with a flexible triangular steel mixer plate in a circular duct flow has been studied in Karlsson et al. (2008). The Reynolds number is 2.55×10^5 at a Mach number of 0.12. The flow induces a static deflection and oscillation of the plate. How this oscillation influences the aero-acoustical properties poses an interesting research question.

Figure 6 Snapshot of the velocity field and flexible mixer plate and pressure field and plate, (a) plate in exhaust system (velocity) (b) plate in exhaust system (pressure) (see online version for colours)



(a)

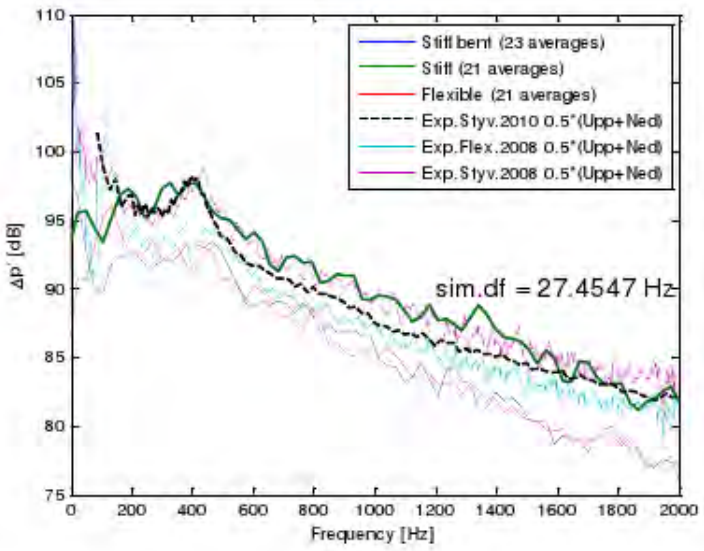


(b)

We set up the duct and flow conditions in Unicorn and introduce a flexible plate. Representative snapshots of the velocity and pressure together with the elastic plate are given in Figure 6. The generated sound spectrum computed from pressure probes in the the duct is given in Figure 7, where we also plot experimental results for comparison. The peak at ca. 400 Hz is captured by the simulation for the stiff case, and for the flexible case the simulation captures the reduction in sound level and has a good match to the experiment for the 300–1,000 Hz frequency band.

This study was performed in collaboration with Andreas Holmberg, Mikael Karlsson, Rodrigo Vilela de Abreu and Mats Åbom at KTH.

Figure 7 Generated sound spectra for Unicorn simulations of stiff, flexible and bent mixer plate, and experiments (see online version for colours)



Notes: Mach 0.207. This study was performed in collaboration with Andreas Holmberg, Mikael Karlsson, Rodrigo Vilela de Abreu and Mats Åbom at KTH.

6.2 Self-oscillating vocal folds

As a step toward building a more complete model of voice production mechanics, we assess the feasibility of a fluid-structure simulation of the vocal fold mechanics in the Unicorn-HPC framework.

In this case we study a geometric model of homogenous silicon rubber vocal folds and a channel together with boundary conditions from an experimental setup given by Becker et al. (2009). The geometric model was kindly provided by Becker. We generate a fluid-structure mesh for the UC framework, and run a parallel simulation.

We apply a constant lung pressure and induce a self-oscillation in the vocal folds. A jet is generated in the small opening between the folds which is periodically cut off when the folds come into contact, see Figure 8. We plot the y-displacement for the

point with the maximum displacement on the vocal folds against time in Figure 9. The frequency is within the range of human phonation (Becker et al., 2009).

Figure 8 Self-oscillating vocal folds: snapshots of geometry and velocity during open and closed phases, and geometry of channel and vocal folds at the top, (a) vocal folds geometry (b) vocal folds closed (c) vocal folds open (d) vocal folds closed (velocity isosurfaces) (e) vocal folds open (velocity isosurfaces) (see online version for colours)

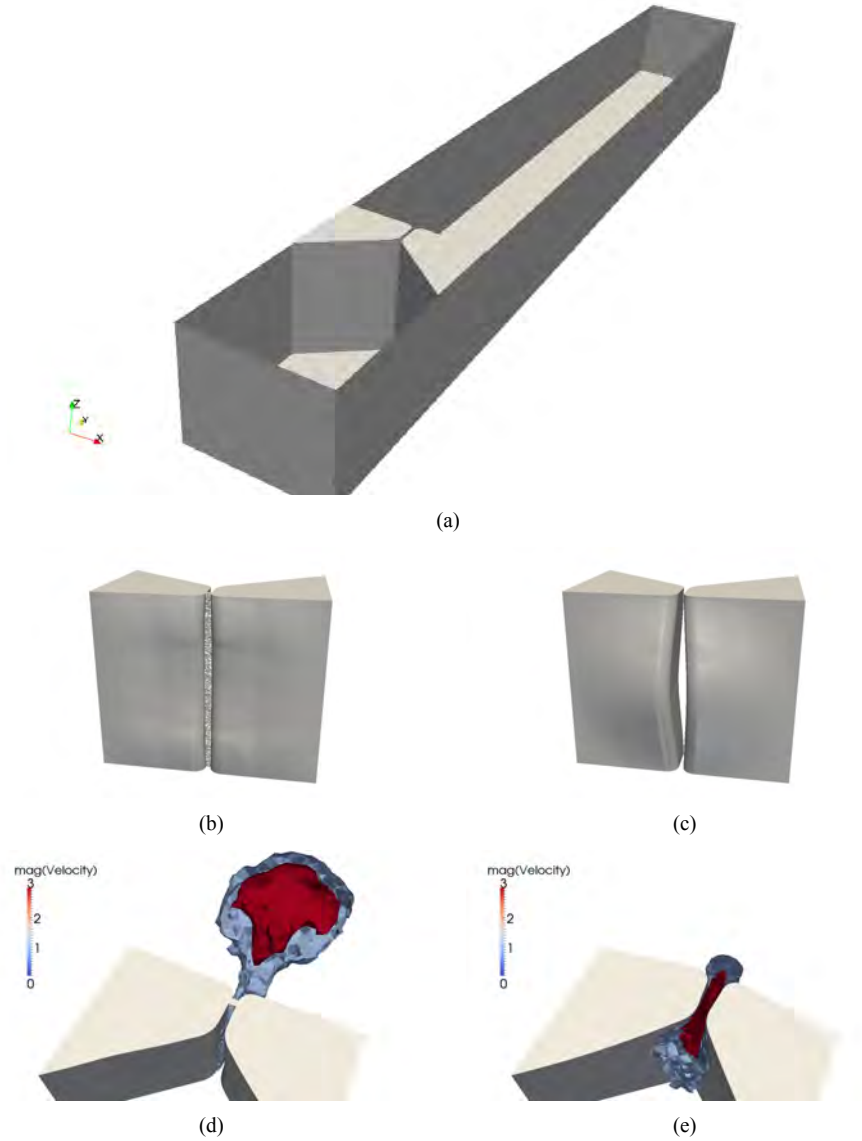
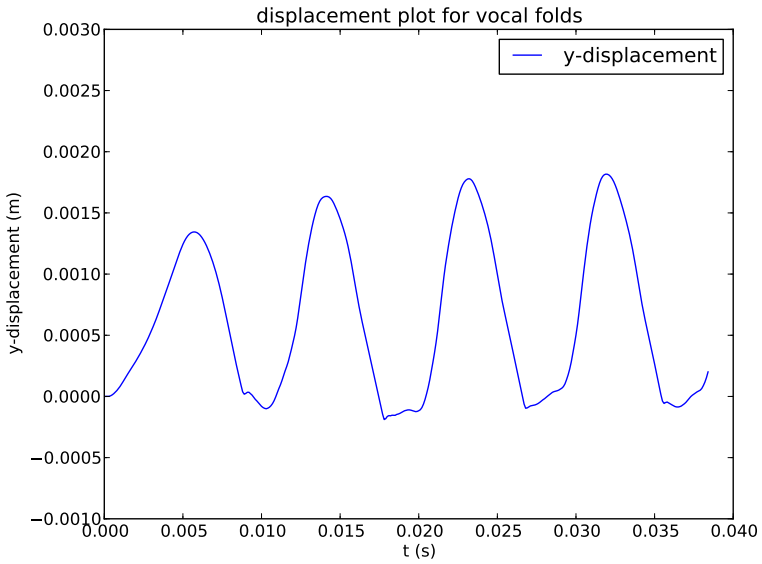


Figure 9 Displacement plot (y-coordinate) for the point with the maximum displacement on the vocal folds (see online version for colours)

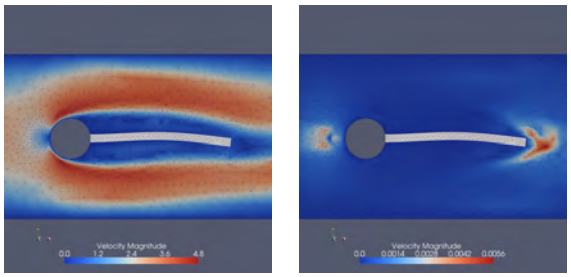


Note: The frequency is within the range of human phonation.

7 Conclusions and future work

In this paper we described the Unicorn-HPC framework allowing FSI simulation using the UC model. We formulated an adaptive method for reaching an error tolerance in a functional of interest by posteriori error estimates using duality arguments for the UC model. We have also presented two industrial and medical 3D fluid-structure applications with favourable results compared to experiments and physiological ranges respectively, which we are targeting for the adaptive method in future work.

Figure 10 (a) Primal velocity for FSI3 benchmark in Turek and Hron (2006) (b) Dual velocity for FSI3 benchmark in Turek and Hron (2006) (see online version for colours)



(a) (b)

We have formulated and solved primal and dual problems and showed convergence and effectiveness for a basic FSI problem by verifying against uniform refinement. Currently we are implementing a parallel version of the adaptive code to be able to deal with more difficult transient problems such as the FSI3 benchmark given in Turek and Hron (2006) and the turbulent 3D problems presented in this paper. Snapshots of the early results with primal and dual solutions are given in Figures 10(a) and 10(b).

Acknowledgements

The authors would like to acknowledge the financial support from the European Research Council, RaySearch Laboratories AB, Swedish Foundation for Strategic Research, the Swedish Research Council, and the Swedish Energy Agency. The simulations were performed on resources provided by the Swedish National Infrastructure for Computing (SNIC) at the National Supercomputer Centre in Sweden (NSC) and PDC – Centre for High-Performance Computing.

We would like to thank Rodrigo Vilela De Abreu from High Performance Computing and Visualization Department at Royal Institute of Technology for collaboration on the numerical results for the flexible mixer plate in an exhaust system, Mats Åbom and Andreas Holmberg from the Marcus Wallenberg Laboratory for Sound and Vibration Research at KTH (MWL) and Mikael Karlsson from Swenox AB for providing experimental results and for valuable discussions. We would also like to thank PD. Dr. Stefan Becker from Friedrich Alexander Universität Erlangen-Nürnberg for sharing the vocal folds geometry.

References

Alnæs, M.S. (2012) 'UFL: a finite element form language', in Logg, A., Mardal, K-A. and Wells, G.N. (Eds.): *Automated Solution of Differential Equations by the Finite Element Method*, Vol. 84, Chapter 17, pp.299–334, Lecture Notes in Computational Science and Engineering, Springer, Berlin, ISBN: 978-3-642-23098-1.

Babuška, I. and Miller, A.D. (1984) 'The post-processing approach in the finite element method, I: calculation of displacements, stresses and other higher derivatives of the dispacements', *Int. J. Numer. Meth. Eng.*, Vol. 20, No. 6, pp.1085–1109.

Babuška, I. (1986) 'Feedback, adaptivity and a posteriori estimates in finite elements: aims, theory, and experience', in I. Babuska, O.C. Zienkiewicz, J. Gago and E.R. de A. Oliveira (Eds.): *Accuracy, Estimates and Adaptive Refinements in Finite Element Computation*, Wiley, New York.

Bazilevs, Y., Calo, V., Zhang, Y. and Hughes, T. (2006) 'Isogeometric fluid-structure interaction analysis with applications to arterial blood flow', *Computational Mechanics*, Vol. 38, Nos. 4–5, pp.310–322.

Bazilevs, Y., Calo, V., Cottrell, J., Hughes, T., Reali, A. and Scovazzi, G. (2007) 'Variational multiscale residual-based turbulence modeling for large eddy simulation of incompressible flows', *Comput. Meth. Appl. Mech. Eng.*, Vol. 197, Nos. 1–4, pp.173–201.

Becker, R. and Rannacher, R. (1996) 'A feed-back approach to error control in adaptive finite element methods: basic analysis and examples', *East-West J. Numer. Math.*, Vol. 4, No. 4, pp.237–264.

Becker, R. and Rannacher, R. (2001) 'An optimal control approach to a posteriori error estimation in finite element methods', *Acta Numerica*, May, Vol. 10, No. 1, pp.1–102, doi:10.1017/S0962492901000010.

Becker, S., Kniesburges, S., Müller, S., Delgado, A., Link, G., Kaltenbacher, M. and Döllinger, M. (2009) 'Flow-structure-acoustic interaction in a human voice model', *Journal of the Acoustical Society of America*, Vol. 125, No. 3, pp.1351–1361.

Compère, G., Remacle, J-F., Jansson, J. and Hoffman, J. (2010) 'A mesh adaptation framework for dealing with large deforming meshes', *Int. J. Numer. Meth. Engng.*, Vol. 82, No. 7, pp.843–867, doi: 10.1002/nme.2788.

Dunne, T. and Rannacher, R. (to appear) 'Adaptive finite element simulation of fluid structure interaction based on an Eulerian variational formulation', in H-J. Bunartz et al. (Eds.): *Fluid-structure Interaction: Modelling, Simulation*, Springer.

Eriksson, K. and Johnson, C. (1988) 'An adaptive finite element method for linear elliptic problems', *Math. Comp.*, Vol. 50, No. 182, pp.361–383.

Eriksson, K., Estep, D., Hansbo, P. and Johnson, C. (1995) 'Introduction to adaptive methods for differential equations', *Acta Numer.*, Vol. 4, No. 1, pp.105–158.

Eriksson, K., Estep, D., Hansbo, P. and Johnson, C. (1996) *Computational Differential Equations*, Cambridge University Press, New York.

Eriksson, K., Estep, D. and Johnson, C. (2003) *Applied Mathematics Body and Soul*, Vol. I–III, Springer-Verlag Publishing, Berlin.

FEniCS (2003) 'Fenics project', in Logg, A., Mardal, K-A. and Wells, G.N. (Eds.): *Automated Solution of Differential Equations by the Finite Element Method*, Vol. 84, Springer, Berlin Heidelberg ISBN: 978-3-642-23098-1.

Giles, M. and Süli, E. (2002) 'Adjoint methods for PDEs: a posteriori error analysis and postprocessing by duality', *Acta Numer.*, Vol. 11, No. 1, pp.145–236.

Guasch, O. and Codina, R. (2007) *A Heuristic Argument for the Sole Use of Numerical Stabilization with No Physical LES Modeling in the Simulation of Incompressible Turbulent Flows*, Preprint Universitat Politecnica de Catalunya.

Guermond, J.L., Pasquetti, R. and Popov, B. (2011) 'From suitable weak solutions to entropy viscosity', *J. Scientific Comput.*, Vol. 49, No. 1, pp.35–50.

Hansbo, P. (2000) 'A Crank-Nicolson type space-time finite element method for computing on moving meshes', *Journal of Computational Physics*, Vol. 159, No. 2, pp.274–289.

Hoffman, J. and Johnson, C. (2007) 'Computational turbulent incompressible flow', Vol. 4 of *Applied Mathematics: Body and Soul*, Springer, Berlin.

Hoffman, J. and Johnson, C. (2008) 'Blow up of incompressible Euler equations', *BIT*, Vol. 48, No. 2, pp.285–307.

Hoffman, J., Jansson, J. and de Abreu, R.V. (2011a) 'Adaptive modeling of turbulent flow with residual based turbulent kinetic energy dissipation', *Comput. Meth. Appl. Mech. Eng.*, Vol. 200, Nos. 37–40, pp.2758–2767.

Hoffman, J., Jansson, J., Jansson, N. and Nazarov, M. (2011b) 'Unicorn: a unified continuum mechanics solver', in *Automated Solutions of Differential Equations by the Finite Element Method*, Springer.

Hoffman, J., Jansson, J. and Stöckli, M. (2011c) 'Unified continuum modeling of fluid-structure interaction', *Math. Mod. Meth. Appl. S.*, Vol. 21, No. 3, pp.491–513.

Hoffman, J., Jansson, J., de Abreu, R.V., Degirmenci, N.C., Jansson, N., Müller, K., Nazarov, M. and Spühler, J.H. (2012a) 'Unicorn: parallel adaptive finite element simulation of turbulent flow and fluid-structure interaction for deforming domains and complex geometry', *Computers and Fluids*, In press.

Hoffman, J., Jansson, J., Degirmenci, C., Jansson, N. and Nazarov, M. (2012b) *Unicorn: A Unified Continuum Mechanics Solver*, Chapter 18, Springer.

Hoffman, J., Jansson, J., Jansson, N., Johnson, C. and de Abreu, R.V. (2012c) *Turbulent Flow and Fluid-structure Interaction*, Chapter 28, Springer.

Hughes, T.J.R. and Brooks, A.N. (1979) 'A multidimensional upwind scheme with no crosswind diffusion, finite element methods for convection dominated flows', in *Applied Mechanics Division (AMD)*, Vol. 34, *Am. Soc. of Mechanical Engineers*.

Hughes, T.J.R. and Brooks, A.N. (1982) *A Theoretical Framework for Petrov-Galerkin Methods, with Discontinuous Weighting Functions: Application to the Streamline Upwind Procedure*, Vol. IV, pp.47–65, John Wiley and Sons Ltd., Chichester.

Jansson, N., Hoffman, J., and Jansson, J. (2012) 'Framework for massively parallel adaptive finite element CFD on tetrahedral meshes', *SIAM J. Sci. Comput.*, Vol. 34, No. 1, pp.C24–C41.

Johnson, C. and Nävert, U. (1981) *An Analysis of Some Finite Element Methods for Advection-diffusion*, North-Holland, Amsterdam.

Karlsson, M., Holmberg, A., Åbom, M., Fallenius, B. and Fransson, J. (2008) 'Experimental determination of the aero-acoustic properties of an in-duct flexible plate', in *Proceedings for 14th AIAA/CEAS Aeroacoustics Conference (29th AIAA Aeroacoustics Conference)*, Vancouver, British Columbia.

Kirby, R.C. (2012) 'FIAT: numerical construction of finite element basis functions', in Logg, A., Mardal, K-A. and Wells, G.N. (Eds.): *Automated Solution of Differential Equations by the Finite Element Method*, Vol. 84, Chapter 13, pp.247–255, Lecture Notes in Computational Science and Engineering, Springer, Berlin, ISBN: 978-3-642-23098-1.

Logg, A., Ølgaard, K.B., Rognes, M.E. and Wells, G.N. (2012) 'FFC: the FEniCS form compiler', in Logg, A., Mardal, K-A. and Wells, G.N. *Automated Solution of Differential Equations by the Finite Element Method*, Vol. 84, Chapter 11, pp.227–238, Lecture Notes in Computational Science and Engineering, Springer, Berlin, ISBN: 978-3-642-23098-1.

Tezduyar, ST.E., Sathe, S. and Stein, K. (2006) 'Solution techniques for the fully-discretized equations in computation of fluid-structure interactions with the space-time formulations', *Computer Methods in Applied Mechanics and Engineering*, Vol. 195, Nos. 41–43, pp.5743–5753.

Turek, S. and Hron, J. (2006) 'A monolithic FEM solver for an ALE formulation of fluid-structure interaction with configuration for numerical benchmarking', in Wesseling, P., Onate, E. and Periaux, J. (Eds.): *Books of Abstracts European Conference on Computational Fluid Dynamics*, p.176, Eccomas CFD 2006.

van der Zee, K., van Brummelen, E., Akkerman, I. and de Borst, R. (2010) 'Goal-oriented error estimation and adaptivity for fluid-structure interaction using exact linearized adjoints', *Computer Methods in Applied Mechanics and Engineering*, in press, corrected proof.

Appendix

Symbolic tool for automatic linearising of UC equations and producing the dual problem in weak form

In this section more explanation about the function $computeDual()$ and its output is provided.

Residual for primal problem

The residual for the primal problem is given as input to $computeDual()$ function. For a different problem these lines are the ones that should be changed.

```
R0 = rho*(U[0].dx(d) + theta*cugradv(U,U[0]))
     -(1-theta)*(cdiv(S0))-theta*(clapl(U[0]))*nu+P.dx(0)
R1 = rho*(U[1].dx(d)+theta*cugradv(U,U[1]))
     -(1-theta)*(cdiv(S1))-theta*(clapl(U[1]))*nu+P.dx(1)
R2 = cdiv(U)
R3 = (1-theta)*(S[0][0].dx(d)-SR[0][0])
R4 = (1-theta)*(S[0][1].dx(d)-SR[0][1])
R5 = (1-theta)*(S[1][0].dx(d)-SR[1][0])
R6 = (1-theta)*(S[1][1].dx(d)-SR[1][1])
R7 = theta.dx(d)+(theta*U[0]).dx(0)+(theta*U[1]).dx(1)
```

To make the notation more compact we have added the differential functions below to the UFL tool (Alnæs, 2012): For vector u, scalar a

$$cugradv(u,a) = \sum_i u_i \frac{\partial a}{\partial x_i}$$

$$clapl(a) = \sum_i \frac{\partial^2 a}{\partial x_i^2}$$

$$cdiv(u) = \sum_i \frac{\partial u[i]}{\partial x_i}$$

In the script U represents velocity, $theta$ phase, P pressure, S solid stress. SR is a variable computed such that

$$SR = \mu * (\nabla U + \nabla U^t) + \nabla U * S + S * \nabla U^t$$

Linearising process

The commands:

```
Rv = [R0, R1, R2, R3, R4, R5, R6,R7]
DDv = [U[0], U[1], P, SS[0], SS[1], SS[2], SS[3], theta]
Jacobian = as_matrix([[linearize(Rv[i],DDv[j])
           for j in range(len(DDv))] for i in range(len(Rv))])
```

creates the jacobian matrix, the command linearise is a recursive function implemented in the same tool. This functionality may be useful for the reader for automatic linearisation of another problem after changing the residual lines.

Dual problem generation

We have seven equations for the primal problem, two for momentum equations in 2D, one for continuity four for solid stress in 2D, and one for phase convection. There will also be seven equations for the dual problem and it is possible to obtain them using the comand:

```
print dualProblem(0)
```

where the first equation is printed as a result.

Starting from the jacobian matrix, the tool performs inner product with a vector of reserved functions, recursively searches for derivatives on variables given in the list named DDV in the tool, does integration by parts if an occurance is found such that result is a boundary term and another inner product where derivative is on the reserved functions. Thus the dual operator which satisfies the bilinear equality (3) is found.

Interpreting output to prepare form file

The command outp(expr) should be read to understand what special terms exist and what they correspond to.

For a 2D problem, for the boundary of a cell of a space-time mesh, the terms n0, n1, n2 correspond to the x, y, time-components of the normal.

- tf0 corresponds to the first test function
- dS corresponds to an integration over the boundaries of cells of the space-time mesh
- dx corresponds to integration in space-time domain
- d/dx0 (ex) or dex/dx0 mean derivative in x direction of expression ex
- d/dx1 (ex) or dex/dx0 mean derivative in y direction of expression ex
- d/dx2 (ex) or dex/dx0 mean derivative in time direction of expression ex
- as already mentioned stabilisation terms may be needed to be added to this results
- for preparing the necessary form files a proper time discretisation is also necessary, where Crank-Nicholson method is used in our form files.

For example the first part of first line of the output of the script:

```
((n1)*tf0*v0*rho*theta*(u1))*dS
```

corresponds to a boundary integration over the boundaries of cells of space-time mesh.

Since we use piecewise constant element for theta and use Crank-Nicholson time stepping in our form files, the terms

```
a = ...+avg(n[1]*vt0*Vm0*rho*theta*Um[1])*dS+...
L = ...+-avg(n[1]*vt0*Vp0*rho*theta*Um[1])*dS+...
```

appear correspondingly in bilinear and linear parts of the form file. The UFL command avg takes average of a value on both sides of a boundary.